OLAF User's Guide and Theory Manual

Kelsey Shaler, Emmanuel Branlard, and Andy Platt

National Renewable Energy Laboratory

NREL is a national laboratory of the U.S. Department of Energy
Office of Energy Efficiency & Renewable Energy
Operated by the Alliance for Sustainable Energy, LLC

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at www.nrel.gov/publications.

Contract No. DE-AC36-08GO28308

Technical Report
NREL/TP-5000-75959
June 2020

OLAF User's Guide and Theory Manual

Kelsey Shaler, Emmanuel Branlard, and Andy Platt

National Renewable Energy Laboratory

Suggested Citation
Shaler, Kelsey, Emmanuel Branlard, and Andy Platt. 2020. *OLAF User's Guide and Theory Manual*. Golden, CO: National Renewable Energy Laboratory. NREL/TP-5000-75959. https://www.nrel.gov/docs/fy20osti/75959.pdf

NREL is a national laboratory of the U.S. Department of Energy
Office of Energy Efficiency & Renewable Energy
Operated by the Alliance for Sustainable Energy, LLC

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at www.nrel.gov/publications.

Contract No. DE-AC36-08GO28308

Technical Report
NREL/TP-5000-75959
June 2020

National Renewable Energy Laboratory
15013 Denver West Parkway
Golden, CO 80401
303-275-3000 • www.nrel.gov

NOTICE

This work was authored by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by the U.S. Department of Energy Office of Energy Efficiency and Renewable Energy Wind Energy Technologies Office. The views expressed herein do not necessarily represent the views of the DOE or the U.S. Government.

Cover Photos by Dennis Schroeder: (clockwise, left to right) NREL 51934, NREL 45897, NREL 42160, NREL 45891, NREL 48097, NREL 46526.

Acknowledgments

This work was authored by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by the U.S. Department of Energy Office of Energy Efficiency and Renewable Energy Wind Energy Technologies Office. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

The authors are also grateful to the Big Adaptive Rotor program for supporting the development of this software.

List of Symbols

BEM	blade-element momentum
CFD	computational fluid dynamics
DOE	U.S. Department of Energy
F_v	core radius factor
t	time
FVW	free vortex wake
N	number of rotor revolutions before wake cutoff condition
$\vec{r}$	vector between point of interest and vortex segment
$\vec{r}(\psi, \zeta)$	position vector of Lagrangian markers
r_c	core radius
r_{c0}	initial core radius
OLAF	cOnvecting LAgrangian Filaments
α	numerical constant = 1.25643
Γ	circulation strength
δ	measure of viscous diffusion
ε	measure of strain
$\Delta\psi$	step size for blade rotation
Ω	rotational speed of wind turbine
ζ	vortex wake age
ζ_0	vortex wake age offset
ν	kinematic viscosity
ψ	azimuth blade position

Table of Contents

List of Figures

List of Tables

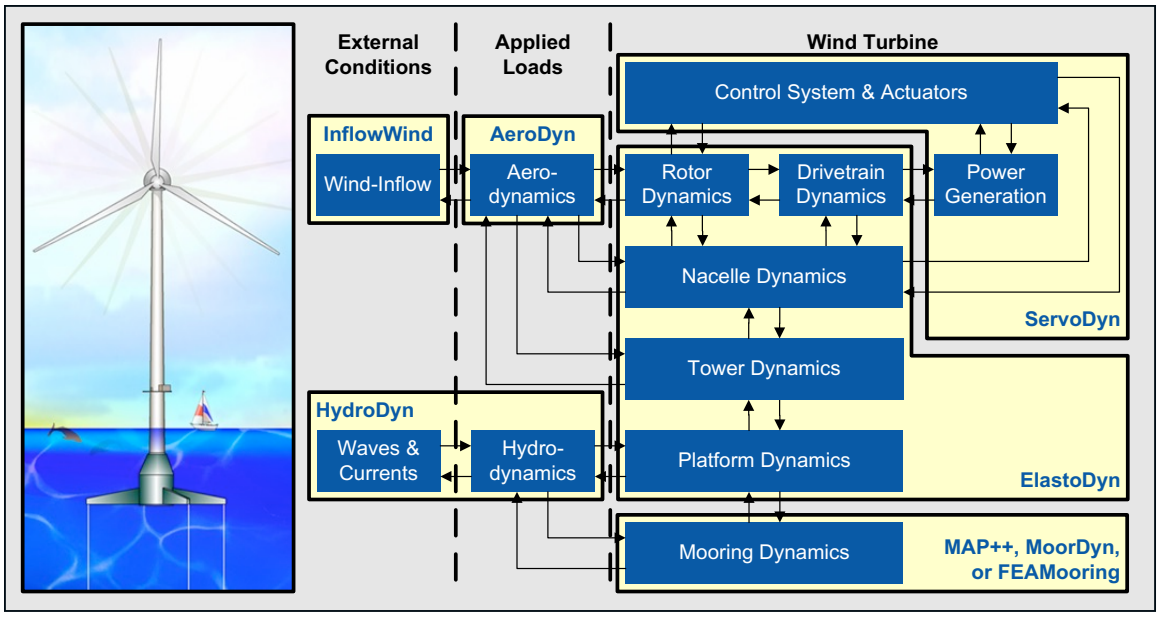
1 Introduction

Over the past few decades, substantial reductions in the cost of wind energy have come from large increases in rotor size. One important consideration for such large turbines is increased blade flexibility. In particular, large blade deflections may lead to a swept area that deviates significantly from the rotor plane. Such deviations violate assumptions used by common aerodynamic models, such as the blade element momentum (BEM) method. Such methods rely on actuator-disk assumptions that are only valid for axisymmetric rotor loads contained in a plane. Large blade deflections may also cause near wake of the turbine to diverge from a uniform helical shape. Further, interactions between turbine blades and the local near wake may increase, thus violating assumptions of models that do not account for the position and dynamics of the near wake. Additionally, highly flexible blades will likely cause increased unsteadiness and three-dimensionality of aerodynamic effects, increasing the importance of accurate and robust dynamic stall models. There are many other complex wind turbine situations that violate simple engineering assumptions. Such situations include obtaining accurate aerodynamic loads for nonstraight blade geometries (e.g., built-in curvature or sweep); skewed flow caused by yawed inflow or turbine tilt; and large rotor motion as a result of placing the turbine atop a compliant offshore floating platform.

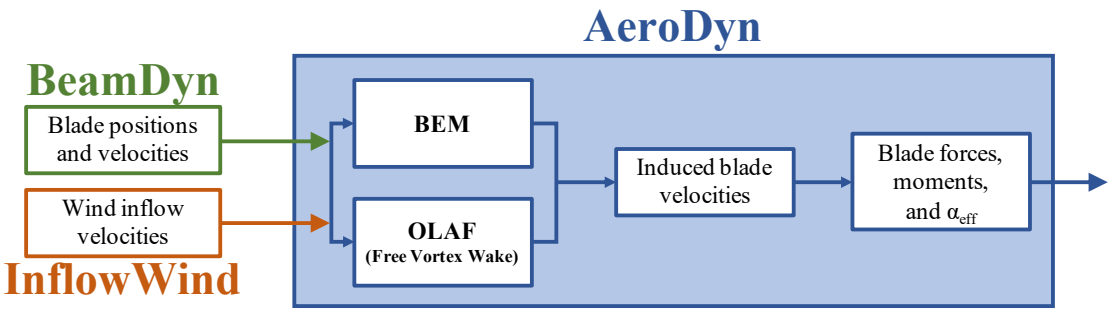
Higher-fidelity aerodynamic models are necessary to account for the increased complexity of flexible and floating rotors. Although computational fluid dynamics (CFD) methods are able to capture such features, their computational cost limits the number of simulations that can be feasibly performed, which is an important consideration in load analysis for turbine design. FVW methods are less computationally expensive than CFD methods while modeling similarly complex physics. As opposed to the BEM methods, FVW methods do not rely on ad-hoc engineering models to account for dynamic inflow, skewed wake, tip losses, or ground effects. These effects are inherently part of the model. Numerous vorticity-based tools have been implemented, ranging from the early treatments by Rosenhead (Rosenhead), the formulation of vortex particle methods by Winckelmans and Leonard (Winckelmans and Leonard), to the recent mixed Eulerian-Lagrangian compressible formulations of Papadakis (Papadakis). Examples of long-standing codes that have been applied in the field of wind energy are GENUVP (Voutsinas), using vortex particles methods, and AWSM (Garrel), using vortex filament methods. Both tools have successfully been coupled to structural solvers. The method was extended by Branlard et al. (Branlard et al.) to consistently use vortex methods to perform aero-elastic simulations of wind turbines in sheared and turbulent inflow. Most formulations rely on a lifting-line representation of the blades, but recently, a viscous-inviscid representation was used in combination with a structural solver (Sessarego et al.).

cOnvecting LAgrangian Filaments (OLAF) is a free vortex wake (FVW) module used to compute the aerodynamic forces on moving two- or three-bladed horizontal-axis wind turbines. This module has been incorporated into the National Renewable Energy Laboratory physics-based engineering tool OpenFAST, which solves the aero-hydro-servo-elastic dynamics of individual wind turbines. OLAF is incorporated into the OpenFAST module *AeroDyn15* as an alternative to the traditional BEM option, as shown in Figure 1. Incorporating the OLAF module within OpenFAST allows for the modeling of highly flexible turbines along with the aero-hydro-servo-elastic response capabilities of OpenFAST. The OLAF module follows the requirements of the OpenFAST modularization framework (Sprague, Jonkman, and Jonkman; Jonkman).

The OLAF module uses a lifting-line representation of the blades, which is characterized by a distribution of bound circulation. The spatial and time variation of the bound circulation results in free vorticity being emitted in the wake. OLAF solves for the turbine wake in a time-accurate manner, which allows the vortices to convect, stretch, and diffuse. The OLAF model is based on a Lagrangian approach, in which the turbine wake is discretized into Lagrangian markers. There are many methods of representing the wake with Lagrangian markers (Branlard). In this work, a hybrid lattice/filament method is used, as depicted in Figure 2. Here, the position of the Lagrangian markers is defined in terms of wake age, ζ, and azimuthal position, ψ. A lattice method is used in the near wake of the blade. The near wake spans over a user-specified angle or distance for nonrotating cases. Though past research has indicated that a near-wake region of 30° is sufficient (Leishman; Ananthan, Leishman, and Ramasamy), it has been shown that a larger near wake is required for high thrust and other challenging conditions. After the near wake region, the wake is assumed to instantaneously roll up into a tip vortex and a root vortex, which are assumed to be the most dominant features



(a) OpenFAST schematic



(b) OLAF and BEM integration with *AeroDyn15*

Figure 1. OpenFAST overview schematic and OLAF integration

for the remainder of the wake (Leishman, Bhagwat, and Bagai). Each Lagrangian marker is connected to adjacent markers by straight-line vortex filaments, approximated to second-order accuracy (Gupta and Leishman). The wake is discretized based on the spanwise location of the blade sections and a specified time step (dt), which may be different from the time step of AeroDyn. After an optional initialization period, the wake is allowed to move and distort, thus changing the wake structure as the markers are convected downstream. To limit computational expense, the root and tip vortices are truncated after a specified distance (***WakeLength***) downstream from the turbine. The wake truncation violates Helmholtz's first law and hence introduces an erroneous boundary condition. To alleviate this, the wake is "frozen" in a buffer zone between a specified buffer distance, ***FreeWakeLength***, and ***WakeLength***. In this buffer zone, the markers convect at the average ambient velocity. In this way, truncation error is minimized (Leishman, Bhagwat, and Bagai). The buffer zone is typically chosen as the convected distance over one rotor revolution.

As part of OpenFAST, induced velocities at the lifting line/blade are transferred to *AeroDyn15* and used to compute the effective blade angle of attack at each blade section, which is then used to compute the aerodynamic forces on the blades. The OLAF method returns the same information as the BEM method, but allows for more accurate

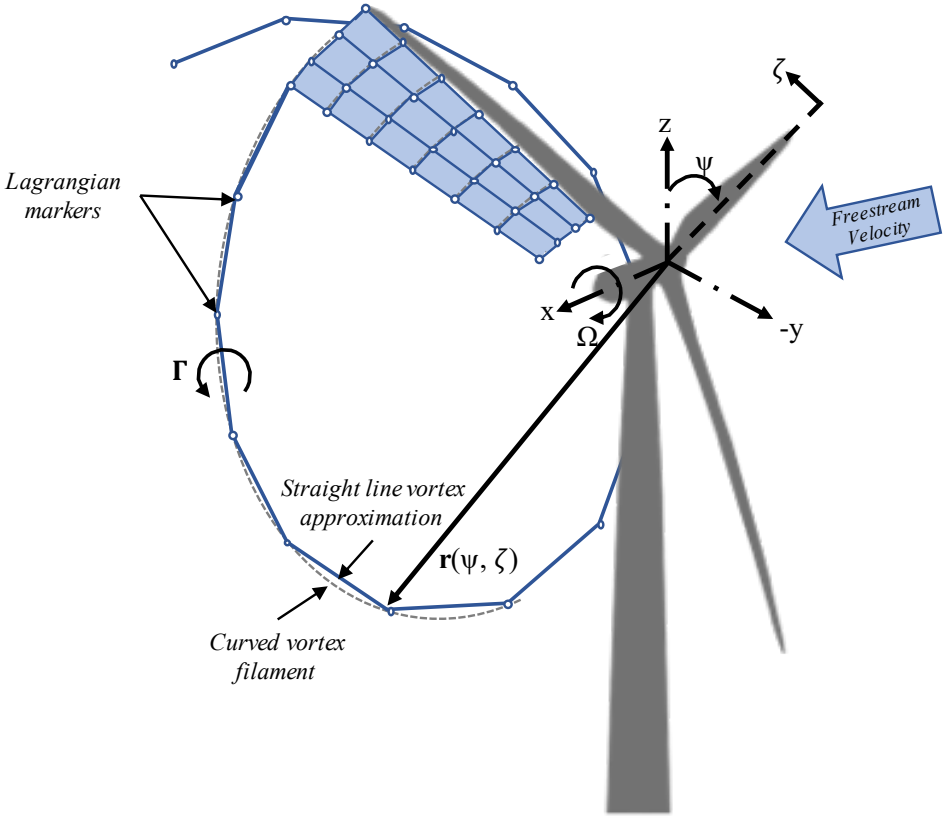


Figure 2. Evolution of near-wake lattice, blade-tip vortex, and Lagrangian markers

calculations in areas where BEM assumptions are violated, such as those discussed above. As the OLAF method is more computationally expensive than BEM, both methods remain available in OpenFAST, and the user may specify in the *AeroDyn15* input file which method is used.

The OLAF input file defines the wake convection and circulation solution methods; wake size and length options; Lagrangian marker regularization (viscous core) method; and other simulation and output parameters. The extents of the near and far wakes are specified by a nondimensional length in terms of rotor diameter. Different regularization functions for the vortex elements are available. Additionally, different methods to compute the regularization parameters of the bound and wake vorticity may be selected. In particular, viscous diffusion may be accounted for by dynamically changing the regularization parameter. Wake visualization output options are also available.

This document is organized as follows. Section 2 covers downloading, compiling, and running OLAF. Section 3 describes the OLAF input file and modifications to the *AeroDyn15* input file. Section 4 details the OLAF output file. Section 5 provides an overview of the OLAF theory, including the free vortex wake method as well as integration into the *AeroDyn15* module. Section 6 presents future work. Example input files and a list of output channels are detailed in Appendices A, B, and C.

2 Running OLAF

As OLAF is a module of OpenFAST, the process of downloading, compiling, and running OLAF is the same as that for OpenFAST. These instructions are available in the OpenFAST documentation.

3 Input Files

No lines should be added or removed from the input files, except in tables where the number of rows is specified.

3.1 Units

OLAF uses the International System of Units (e.g., kg, m, s, N). Angles are assumed to be in degrees unless otherwise specified.

3.2 OLAF Primary Input File

The primary OLAF input file defines general free wake options, circulation model selection and specification, near- and far-wake length, and wake visualization options. Each section within the file corresponds to an aspect of the OLAF model. For most parameters, the user may specify the value "default" (with or without quotes), in which case a default value, defined below, is used by the program.

See Appendix A for a sample OLAF primary input file.

3.2.1 General Options

IntMethod [switch] specifies which integration method will be used to convect the Lagrangian markers. There are four options: 1) fourth-order Runge-Kutta ***[1]***, 2) fourth-order Adams-Bashforth ***[2]***, 3) fourth-order Adams-Bashforth-Moulton ***[3]***, and 4) first-order forward Euler ***[5]***. The default option is ***[5]***. These methods are specified in Section 5.4.

DTfvw [sec] specifies the time interval at which the module will update the wake. The time interval must be a multiple of the time step used by *AeroDyn15*. The blade circulation is updated at each intermediate time step based on the intermediate blades positions and wind velocities. The default value is dt_{aero}, where dt_{aero} is the time step used by AeroDyn.

FreeWakeStart [sec] specifies at what time the wake evolution is classified as "free." Before this point, the Lagrangian markers are simply convected with the freestream velocity. After this point, induced velocities are computed and affect the marker convection. If a time less than or equal to zero is given, the wake is "free" from the beginning of the simulation. The default value is 0.

FullCircStart [sec] specifies at what time the blade circulation reaches full strength. If this value is specified to be > 0, the circulation is multiplied by a factor of 0 at $t = 0$ and linearly increasing to a factor of 1 for $t >$***FullCircStart***. The default value is 0.

3.2.2 Circulation Specifications

CircSolvMethod [switch] specifies which circulation method is used. There are three options: 1) C_l-based iterative procedure ***[1]***, 2) no-flow through ***[2]***, and 3) prescribed ***[3]***. The default option is ***[1]***. These methods are described in Section 5.3.

CircSolvConvCrit [-] specifies the dimensionless convergence criteria used for solving the circulation. This variable is only used if ***CircSolvMethod = [1]***. The default value is 0.001, corresponding to 0.1% error in the circulation between two iterations.

CircSolvRelaxation [-] specifies the relaxation factor used to solve the circulation. This variable is only used if ***CircSolvMethod = [1]***. The default value is 0.1.

CircSolvMaxIter [-] specifies the maximum number of iterations used to solve the circulation. This variable is only used if ***CircSolvMethod = [1]***. The default value is 30.

PrescribedCircFile [quoted string] specifies the file containing the prescribed blade circulation. This option is only used if ***CircSolvMethod = [3]***. The circulation file format is a delimited file with one header line and two columns. The first column is the dimensionless radial position [r/R]; the second column is the bound circulation value in [m^2/s].

The radial positions do not need to match the AeroDyn node locations. A sample prescribed circulation file is given in Appendix B.

3.2.3 Wake Extent and Discretization Options

nNWPanel [-] specifies the number of FVW time steps (***DTfvw***) for which the near-wake lattice is computed. In the future, this value will be defined as an azimuthal span in degrees or a downstream distance in rotor diameter.

WakeLength [D] specifies the length, in rotor diameters, of the far wake. The default value is 8.[1]

FreeWakeLength [D] specifies the length, in rotor diameters, for which the turbine wake is convected as "free." If ***FreeWakeLength*** is greater than ***WakeLength***, then the entire wake is free. Otherwise, the Lagrangian markers located within the buffer zone delimited by ***FreeWakeLength*** and ***WakeLength*** are convected with the average velocity. The default value is 6.[2]

FWShedVorticity [flag] specifies whether shed vorticity is included in the far wake. The default option is ***[False]***, specifying that the far wake consists only of the trailed vorticity from the root and tip vortices.

3.2.4 Wake Regularization and Diffusion Options

DiffusionMethod [switch] specifies which diffusion method is used to account for viscous diffusion. There are two options: 1) no diffusion ***[0]*** and 2) the core-spreading method ***[1]***. The default option is ***[0]***.

RegDetMethod [switch] specifies which method is used to determine the regularization parameters. There are two options: 1) manual ***[0]*** and 2) optimized ***[1]***. The manual option requires the user to specify the parameters listed in this subsection. The optimized option determines the parameters for the user. The default option is ***[0]***.

RegFunction [switch] specifies the regularization function used to remove the singularity of the vortex elements, as specified in Section 5.4. There are five options: 1) no correction ***[0]***, 2) the Rankine method ***[1]***, 3) the Lamb-Oseen method ***[2]***, 4) the Vatistas method ***[3]***, and 5) the denominator offset method ***[4]***. The functions are given in Section 5.7.3. The default option is ***[3]***.

WakeRegMethod [switch] specifies the method of determining viscous core radius (i.e., the regularization parameter). There are four options: 1) constant ***[1]***, 2) stretching ***[2]***, 3) age ***[3]***, and 4) stretching and age ***[4]***. The methods are described in Section 5.7.4. The default option is ***[1]***.

WakeRegParam [m] specifies the wake regularization parameter, which is the regularization value used at the initialization of a vortex element. If the regularization method is "constant", this value is used throughout the wake.

BladeRegParam [m] specifies the bound vorticity regularization parameter, which is the regularization value used for the vorticity elements bound to the blades.

CoreSpreadEddyVisc [-] specifies the eddy viscosity parameter δ. The parameter is used for the core-spreading method (***DiffusionMethod=[1]***) and the regularization method with age (***WakeRegMethod=[3]***). The variable δ is described in Section 5.7.4. The default value is 100.

3.2.5 Wake Treatment Options

TwrShadowOnWake [flag] specifies whether the tower potential flow and tower shadow have an influence on the wake convection. The tower shadow model, when activated in AeroDyn, always has an influence on the lifting line, hence the induction and loads on the blade. This option only concerns the wake. The default option is ***[False]***.

ShearVorticityModel [switch] specifies whether shear vorticity is modeled in addition to the sheared inflow prescribed by *InflowWind*. There are two options: 1) no treatment ***[0]*** and 2) mirrored vorticity ***[1]***. The mirrored vorticity accounts for the ground effect. Dedicated options to account for the shear vorticity will be implemented at a later time. The shear velocity profile is handled by *InflowWind* irrespective of this input. The default option is ***[0]***.

[1] At present, this variable is called nFWPanel and specified as the number of far wake panels. This will be changed soon.
[2] At present, this variable is called nFWPanelFree and specified as the number of free far wake panels. This will be changed soon.

3.2.6 Speedup Options

VelocityMethod [switch] specifies the method used to determine the velocity. There are two options: 1) Biot-Savart law applied to the vortex segments ***[1]*** and 2) tree formulation using a particle representation ***[2]***. The default option is ***[1]***.

TreeBranchFactor [-] specifies the dimensionless distance, in branch radius, above which a multipole calculation is used instead of a direct evaluation. This option is only used in conjunction with the tree code (***VelocityMethod=[2]***).

PartPerSegment [-] specifies the number of particles that are used when a vortex segment is represented by vortex particles. The default value is 1.

3.2.7 Output Options

WrVTK [flag] specifies if Visualization Toolkit (VTK) visualization files are to be written out. ***WrVTK=[0]*** does not write out any VTK files. ***WrVTK=[1]*** outputs a VTK file at every time step. The outputs are written in the folder, `vtk_fvw`. The parameters ***WrVTK***, ***VTKCoord***, and ***VTK_fps*** are independent of the glue code VTK output options.

VTKBlades [-] specifies how many blade VTK files are to be written out. ***VTKBlades***$= n$ outputs VTK files for n blades, with 0 being an acceptable value. The default value is 1.

VTKCoord [switch] specifies the coordinate system in which the VTK files are written. There are two options: 1) global coordinate system ***[1]*** and 2) hub coordinate system ***[2]***. The default option is ***[1]***.

VTK_fps [1/sec] specifies the output frequency of the VTK files. The provided value is rounded to the nearest allowable multiple of the time step. The default value is $1/dt_{\text{fvw}}$. Specifying ***VTK_fps=[all]*** is equivalent to using the value $1/dt_{\text{aero}}$.

3.3 AeroDyn15 Input File

3.3.1 Input file modifications

As OLAF is incorporated into the *AeroDyn15* module, a wake computation option has been added to the *AeroDyn15* input file and a line has been added. These additions are as follows:

WakeMod specifies the type of wake model that is used. ***WakeMod=[3]*** has been added to allow the user to switch from the traditional BEM method to the OLAF method.

FVWFile [string] specifies the OLAF module file. The path is relative to the AeroDyn file, unless an absolute path is provided.

3.3.2 Relevant sections

The BEM options (e.g. tip-loss, skew, and dynamic models) are read and discarded when ***WakeMod=[3]***. The following sections and parameters remain relevant and are used by the vortex code:

- general options (e.g., airfoil and tower modeling);
- environmental conditions;
- dynamic stall model options;
- airfoil and blade information;
- tower aerodynamics; and
- outputs.

4 Output Files

The OLAF module itself does not produce its own output file. However, additional output channels are made available in *AeroDyn15*. As such, the *AeroDyn15* output file is briefly described as well as the outputs made available with OLAF. Visualization files are generated by using the parameter ***WrVTK***. This parameter is available in the OLAF input file, in which case the VTK files are written to the folder `vtk_fvw`, or the primary `.fst` file, in which case the VTK files are written to the folder `vtk`.

4.1 Results File

OpenFAST generates a master results file that includes the *AeroDyn15* results. The results are in table format, where each column is a data channel, and each row corresponds to a simulation-output time step. The data channels are specified in the *OUTPUTS* section in the *AeroDyn15* primary input file. The column format of the AeroDyn-generated files is specified using the ***OutFmt*** parameter of the OpenFAST driver input file.

5 OLAF Theory

This section details the OLAF method and provides an overview of the computational method, followed by a brief explanation of its integration with OpenFAST.

5.1 Introduction - Vorticity Formulation

The vorticity equation for incompressible homogeneous flows in the absence of non-conservative force is given by Eq. 5.1

$$\frac{d\vec{\omega}}{dt} = \frac{\partial \vec{\omega}}{\partial t} + \underbrace{(\vec{u}\cdot\nabla)}_{\text{convection}}\ \vec{\omega} = \underbrace{(\vec{\omega}\cdot\nabla)\vec{u}}_{\text{strain}} + \underbrace{\nu\Delta\vec{\omega}}_{\text{diffusion}} \tag{5.1}$$

Here, $\vec{\omega}$ is the vorticity, $\vec{u}$ is the velocity, and ν is the viscosity. In free vortex wake methods, the vorticity equation is used to describe the evolution of the wake vorticity. Different approximations are introduced to ease its resolution, such as projecting the vorticity onto a discrete number of vortex elements (here vortex filaments), and separately treating the convection and diffusion steps, known as viscous-splitting. Several complications arise from the method; in particular, the discretization requires a regularization of the vorticity field (or velocity field) to ensure a smooth approximation.

The forces exerted by the blades onto the flow are expressed in vorticity formulation as well. This vorticity is bound to the blade and has a circulation associated with the lift force. A lifting-line formulation is used here to model the bound vorticity.

The different models of the implemented free vortex code are described in the following sections.

5.2 Discretization - Projection

The numerical method uses a finite number of states to model the continuous vorticity distribution. To achieve this, the vorticity distribution is projected onto a basis function, which is referred to as vortex elements. Vortex filaments are used as elements that represent the vorticity field. A vortex filament is delimited by two points and hence assumes a direction formed by these two points. A vorticity tube is oriented along the unit vector $\vec{e}_x$ of cross section dS and length l. It can then be approximated by a vortex filament of length l oriented along the same direction. The total vorticity of the tube and the vortex filaments are the same and related by:

$$\vec{\omega}\, dS = \vec{\Gamma} \tag{5.2}$$

where $\vec{\Gamma}$ is the circulation intensity of the vortex filament. If the vorticity tubes are complex and occupy a large volume, projection onto vortex filaments is difficult and projection onto vortex particles is more appropriate. Assuming the wake is confined to a thin vorticity layer defining a velocity jump of known direction, it is possible to approximate the wake vorticity sheet as a mesh of vortex filaments. This is the basis of vortex filament wake methods. Vortex filaments are a singular representation of the vorticity field, as they occupy a line instead of a volume. To better represent the vorticity field, the filaments are "inflated", a process referred to as regularization (see Section 5.7). The regularization of the vorticity field also regularizes the velocity field and avoids the singularities that would otherwise occur.

5.3 Lifting-Line Representation

The code relies on a lifting-line formulation to model the blades. Lifting-line methods effectively lump the loads at each cross-section of the blade onto the mean line of the blade and do not account for the geometry of each cross-section. In the vorticity-based version of the lifting-line method, the blade is represented by a line of varying circulation. The line follows the motion of the blade and is referred to as "bound" circulation. The bound circulation does not follow the same dynamic equation as the free vorticity of the wake. Instead, the intensity is linked to airfoil lift via the Kutta-Joukowski theorem. Spanwise variation of the bound circulation results in vorticity being emitted into the

wake. This is referred to as "trailed vorticity". Time changes of the bound circulation are also emitted in the wake and referred to as "shed" vorticity. The remaining subsections describe the representation of the bound vorticity in greater detail.

5.3.1 Lifting-Line Panels and Emitted Wake Panels

The lifting-line and wake representation is illustrated in Fig. 3. The blade lifting-line is discretized into a finite number of panels, each of them forming a four sided vortex ring. The spanwise discretization follows the discretization of the AeroDyn blade input file. The number of spanwise panels, n_{LL}, is one less than the total number of AeroDyn nodes, ***NumBlNds***. The sides of the panels coincide with the lifting-line and the trailing edge of the blade. The lifting-line is currently defined as the 1/4 chord location from the leading edge (LE). More details on the panelling are provided in Section 5.3.2. At a given time step, the circulation of each lifting-line panel is determined according to one of the three methods developed in Section 5.3.3. At the end of the time step, the circulation of each lifting-line panel is emitted into the wake, forming free vorticity panels. To satisfy the Kutta condition, the circulation of the first near wake panel and the bound circulation are equivalent (see Fig. 3b). The wake panels model the thin shear layer resulting from the continuation of the blade boundary layer. This shear layer can be modeled using a continuous distribution of vortex doublets. A constant doublet strength is assumed on each panel, which in turn is equivalent to a vortex ring of constant circulation. The current implementation stores the positions and circulations of the panel corner points. In

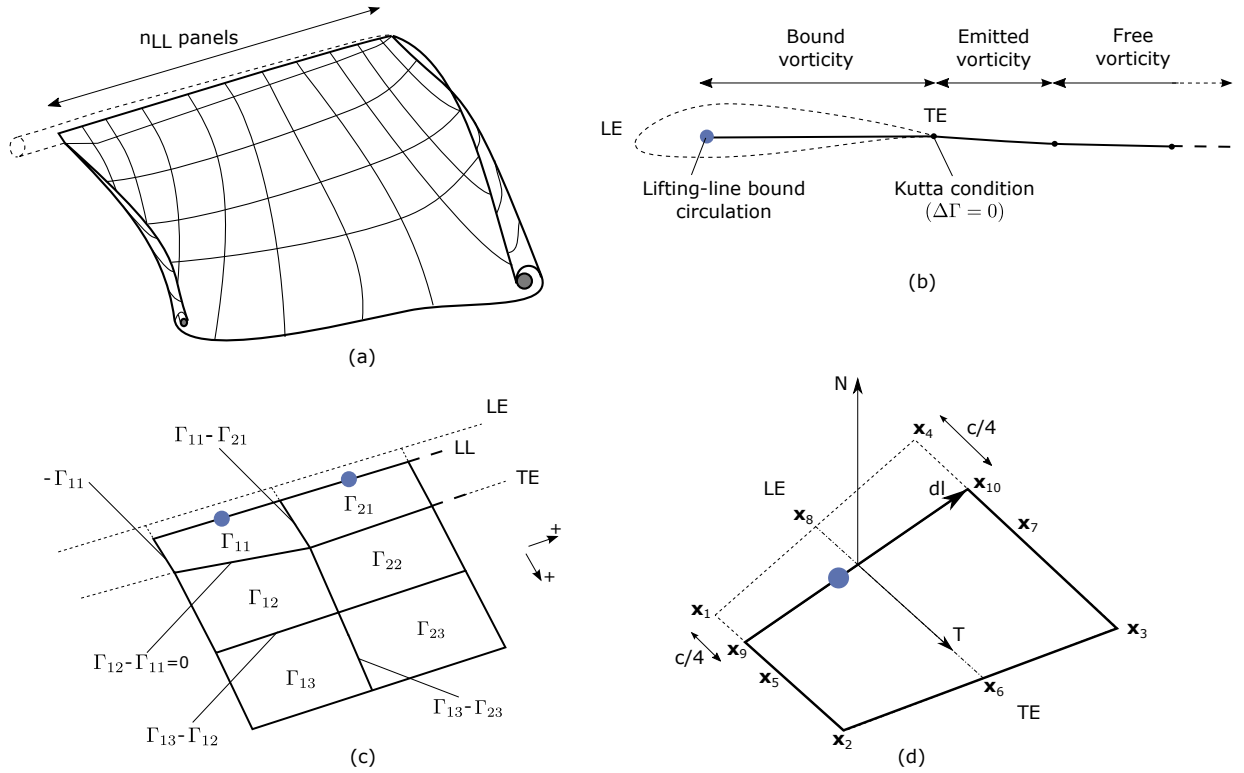


Figure 3. Wake and lifting-line vorticity discretized into vortex ring panels. (a) Overview. (b) Cross-sectional view, defining the leading-edge, trailing edge, and lifting-line. (c) Circulation of panels and corresponding circulation for vorticity segments between panels. (d) Geometrical quantities for a lifting-line panel.

the vortex ring formulation, the boundary between two panels corresponds to a vortex segment of intensity equal to the difference of circulation between the two panels. The convention used to define the segment intensity based on the panels intensity is shown in Fig. 3c. Since the circulation of the bound panels and the first row of near wake panels are equal, the vortex segments located on the trailing edge have no circulation.

5.3.2 Panelling

The definitions used for the panelling of the blade are given in Fig. 3d, following the notations of van Garrel (Garrel). The LE and trailing edge (TE) locations are directly obtained from the AeroDyn mesh. The LE and TE define the corner points at two spanwise locations: $\vec{x}_1$, $\vec{x}_2$, $\vec{x}_3$, and $\vec{x}_4$. The current implementation assumes that the aerodynamic center, the lifting-line, and the 1/4 chord location all coincide. For a given panel, the lifting-line is then delimited by the points $\vec{x}_9 = 3/4\vec{x}_1 + 1/4\vec{x}_2$ and $\vec{x}_{10} = 3/4\vec{x}_4 + 1/4\vec{x}_3$. The mid points of the four panel sides are noted $\vec{x}_5$, $\vec{x}_6$, $\vec{x}_7$, and $\vec{x}_8$. The lifting-line vector ($\vec{dl}$) as well as the vectors tangential ($\vec{T}$) and normal ($\vec{N}$) to the panel are defined as:

$$\vec{dl} = \vec{x}_{10} - \vec{x}_9, \qquad \vec{T} = \frac{\vec{x}_6 - \vec{x}_8}{|\vec{x}_6 - \vec{x}_8|}, \qquad \vec{N} = \frac{\vec{T} \times \vec{dl}}{|\vec{T} \times \vec{dl}|} \tag{5.3}$$

The area of the panel is obtained as $dA = |(\vec{x}_6 - \vec{x}_8) \times (\vec{x}_7 - \vec{x}_5)|$. For ***CircSolvMethod=[1]***, the control points are located on the lifting-line at the location $\vec{x}_9 + \eta_j \vec{dl}$. The factor η_j is determined based on the full-cosine approximation of van Garrel. This is based on the spanwise widths of the current panel, w_j, and the neighboring panels w_{j-1} and w_{j+1}:

$$\eta_j = \frac{1}{4}\left[\frac{w_{j-1}}{w_{j-1}+w_j} + \frac{w_j}{w_j+w_{j+1}} + 1\right], \; j = 2..n-1, \quad \eta_1 = \frac{w_1}{w_1+w_2}, \quad \eta_n = \frac{w_{n-1}}{w_{n-1}+w_n} \tag{5.4}$$

For an equidistant spacing, this discretization places the control points at the middle of the lifting-line ($\eta = 0.5$). Theoretical circulation results for an elliptic wing with a cosine spacing are retrieved with such discretization since it places the control points closer to stronger trailing segments at the wing extremities (e.g., Kerwin).

5.3.3 Circulation Solving Methods

Three methods are implemented to determine the bound circulation strength. They are selected using the input ***CircSolvMethod***, and are presented in the following sections.

5.3.3.1 Cl-Based Iterative Method

The Cl-based iterative method determines the circulation within a nonlinear iterative solver that utilizes the polar data at each control point on the lifting line. The algorithm ensures that the lift obtained using the angle of attack and the polar data matches the lift obtained with the Kutta-Joukowski theorem. At present, it is the preferred method to compute circulation along the blade span. It is selected with ***CircSolvMethod=[1]***. The method is described in the work from van Garrel (Garrel). The algorithm is implemented in an iterative approach using the following steps:

1. The circulation distribution from the previous time step is used as a guessed circulation, Γ_{prev}.
2. The velocity at each control point j is computed as the sum of the wind velocity, the structural velocity, and the velocity induced by all the vorticity in the domain, evaluated at the control point location.

$$\vec{v}_j = \vec{V}_0 - \vec{V}_{\text{elast}} + \vec{v}_{\omega,\text{free}} + \vec{v}_{\Gamma_{ll}} \tag{5.5}$$

   $\vec{v}_{\omega,\text{free}}$ is the velocity induced by all free vortex filaments, as introduced in Eq. 5.16. The contribution of $\vec{v}_{\Gamma_{ll}}$ comes from the lifting-line panels and the first row of near wake panels, for which the circulation is set to Γ_{prev}
3. The circulation for all lifting-line panels j is obtained as follows.

$$\Gamma_{ll,j} = \frac{1}{2} C_{l,j}(\alpha_j) \frac{\left[(\vec{v}_j \cdot \vec{N})^2 + (\vec{v}_j \cdot \vec{T})^2\right]^2 dA}{\sqrt{\left[(\vec{v}_j \times \vec{dl}) \cdot \vec{N}\right]^2 + \left[(\vec{v}_j \times \vec{dl}) \cdot \vec{T}\right]^2}}, \quad \text{with} \quad \alpha_j = \text{atan}\left(\frac{\vec{v}_j \cdot \vec{N}}{\vec{v}_j \cdot \vec{T}}\right) \tag{5.6}$$

   The function $C_{l,j}$ is the lift coefficient obtained from the polar data of blade section j and α_j is the angle of attack at the control point.

4. The new circulation is set using the relaxation factor k_{relax} (***CircSolvRelaxation***):

$$\Gamma_{\text{new}} = \Gamma_{\text{prev}} + k_{\text{relax}}\Delta\Gamma, \qquad \Delta\Gamma = \Gamma_{ll} - \Gamma_{\text{prev}} \tag{5.7}$$

5. Convergence is checked using the criterion k_{crit} (***CircSolvConvCrit***):

$$\frac{\max(|\Delta\Gamma|}{\text{mean}(|\Gamma_{\text{new}}|)} < k_{\text{crit}} \tag{5.8}$$

If convergence is not reached, steps 2-5 are repeated using Γ_{new} as the guessed circulation Γ_{prev}.

5.3.3.2 No-flow-through Method

A Weissinger-L-based representation (Weissinger) of the lifting surface is also available (Bagai and Leishman; Gupta; Ribera). In this method, the circulation is solved by satisfying a no-flow through condition at the 1/4-chord points. It is selected with ***CircSolvMethod=[2]***.

5.3.3.3 Prescribed Circulation

The final available method prescribes a constant circulation. A user specified spanwise distribution of circulation is prescribed onto the blades. It is selected with ***CircSolvMethod=[3]***.

5.4 Free Vorticity Convection

The governing equation of motion for a vortex filament is given by the convection equation of a Lagrangian marker:

$$\frac{d\vec{r}}{dt} = \vec{V}(\vec{r},t) \tag{5.9}$$

where $\vec{r}$ is the position of a Lagrangian marker. The Lagrangian markers are the end points of the vortex filaments. The Lagrangian convection of the filaments stretches the filaments and thus automatically accounts for strain in the vorticity equation.

At present, a first-order forward Euler method is used to numerically solve the left-hand side of Eq. 5.9 for the vortex filament location (***IntMethod=[5]***). This is an explicit method solved using Eq. 5.10.

$$\vec{r} = \vec{r} + \vec{V}\Delta t \tag{5.10}$$

5.5 Free Vorticity Convection in Polar Coordinates

The governing equation of motion for a vortex filament is given by:

$$\frac{d\vec{r}(\psi,\zeta)}{dt} = \vec{V}[\vec{r}(\psi,\zeta),t] \tag{5.11}$$

Using the chain rule, Eq. 5.11 is rewritten as:

$$\frac{\partial\vec{r}(\psi,\zeta)}{\partial\psi} + \frac{\partial\vec{r}(\psi,\zeta)}{\partial\zeta} = \frac{\vec{V}[\vec{r}(\psi,\zeta),t]}{\Omega} \tag{5.12}$$

where $d\psi/dt = \Omega$ and $d\psi = d\zeta$ (Leishman, Bhagwat, and Bagai). Here, $\vec{r}(\psi,\zeta)$ is the position vector of a Lagrangian marker, and $\vec{V}[\vec{r}(\psi,\zeta)]$ is the velocity.

5.6 Induced Velocity and Velocity Field

The velocity term on the right-hand side of Eq. 5.9 is a nonlinear function of the vortex position, representing a combination of the freestream and induced velocities (Hansen). The induced velocities at point $\vec{x}$, caused by each straight-line filament, are computed using the Biot-Savart law, which considers the locations of the Lagrangian markers and the intensity of the vortex elements (Leishman, Bhagwat, and Bagai):

$$d\vec{v}(\vec{x}) = \frac{\Gamma}{4\pi}\frac{d\vec{l}\times\vec{r}}{r^3} \tag{5.13}$$

Here, Γ is the circulation strength of the filament, $\vec{dl}$ is an elementary length along the filament, $\vec{r}$ is the vector between a point on the filament and the control point $\vec{x}$, and $r = |\vec{r}|$ is the norm of the vector. The integration of the Biot-Savart law along the filament length, delimited by the points $\vec{x}_1$ and $\vec{x}_2$ leads to:

$$\vec{v}(\vec{x}) = F_v\frac{\Gamma}{4\pi}\frac{(r_1+r_2)}{r_1 r_2(r_1 r_2+\vec{r}_1\cdot\vec{r}_2)}\vec{r}_1\times\vec{r}_2 \tag{5.14}$$

with $\vec{r}_1 = \vec{x}-\vec{x}_1$ and $\vec{r}_2 = \vec{x}-\vec{x}_2$. The factor F_v is a regularization parameter, discussed in Section 5.7.3. r_0 is the filament length, where $\vec{r}_0 = \vec{x}_2-\vec{x}_1$. The distance orthogonal to the filament is:

$$\rho = \frac{|\vec{r}_1\times\vec{r}_2|}{r_0} \tag{5.15}$$

The velocity at any point of the domain is obtained by superposition of the velocity induced by all vortex filaments, and by superposition of the primary flow, $\vec{V}_0$, (here assumed divergence free):

$$\vec{V}(\vec{x}) = \vec{V}_0(\vec{x}) + \vec{v}_\omega(\vec{x}), \quad \text{with} \quad \vec{v}_\omega(\vec{x}) = \sum_k \vec{v}_k(\vec{x}) \tag{5.16}$$

where the sum is over all the vortex filaments, each of intensity Γ_k. The intensity of each filament is determined by spanwise and time changes of the bound circulation, as discussed in Section 5.3. In tree-based methods, the sum over all vortex elements is reduced by lumping together the elements that are far away from the control points.

5.7 Regularization

5.7.1 Regularization and viscous diffusion

The singularity that occurs in Eq. 5.13 greatly affects the numerical accuracy of vortex methods. By regularizing the "1-over-r" kernel of the Biot-Savart law, it is possible to obtain a numerical method that converges to the Navier-Stokes equations. The regularization is used to improve the regularity of the discrete vorticity field, as compared to the "true" continuous vorticity field. This regularization is usually obtained by convolution with a smooth function. In this case, the regularization of the vorticity field and the velocity field are the same. Some engineering models also perform regularization by directly introducing additional terms in the denominator of the Biot-Savart velocity kernel. The factor, F_v, was introduced in Eq. 5.14 to account for this regularization.

In the convergence proofs of vortex methods, regularization and viscous diffusion are two distinct aspects. It is common practice in vortex filament methods to blur the notion of regularization with the notion of viscous diffusion. Indeed, for a physical vortex filament, viscous effects prevent the singularity from occurring and diffuse the vortex strength with time. The circular zone where the velocity drops to zero around the vortex is referred to as the vortex core. A length increase of the vortex segment will result in a vortex core radius decrease, and vice versa. Diffusion, on the other hand, continually spreads the vortex radially.

Because of the previously mentioned analogy, practitioners of vortex filament methods often refer to regularization as "viscous-core" models and regularization parameters as "core-radii." Additionally, viscous diffusion is often introduced by modifying the regularization parameter in space and time instead of solving the diffusion from the vorticity equation. The distinction is made explicit in this document when clarification is required, but a loose terminology is used when the context is clear.

5.7.2 Determination of the regularization parameter

The regularization parameter is both a function of the physics being modeled (blade boundary layer and wake) and the choice of discretization. Contributing factors are the chord length, the boundary layer height, and the volume that each vortex filament is approximating. Currently the choice is left to the user (***RegDetMethod=[0]***). Empirical results for a rotating blade are found in the work of Gupta (Gupta). As a guideline, the regularization parameter may be chosen as twice the average spanwise discretization of the blade. This guideline is implemented when the user chooses ***RegDetMethod=[1]***. Further refinement of this option will be considered in the future.

5.7.3 Implemented regularization functions

Several regularization functions have been developed (Rankine; Scully; Vatistas, Koezel, and Mih). At present, five options are available: 1) No correction, 2) the Rankine method, 3) the Lamb-Oseen method, 4) the Vatistas method, and 5) the denominator offset method. If no correction method is used, (***RegFunction=[0]***), $F_\nu = 1$. The remaining methods are detailed in the following sections. Here, r_c is the regularization parameter (***WakeRegParam***) and ρ is the distance to the filament. Both variables are expressed in meters.

5.7.3.1 Rankine

The Rankine method (Rankine) is the simplest regularization model. With this method, the Rankine vortex has a finite core with a solid body rotation near the vortex center and a potential vortex away from the center. If this method is used (***RegFunction=[1]***), the viscous core correction is given by Eq. 5.17.

$$F_\nu = \begin{cases} \rho^2/r_c^2 & 0 < \rho < 1 \\ 1 & \rho > 1 \end{cases} \tag{5.17}$$

Here, r_c is the viscous core radius of a vortex filament, detailed in Section 5.7.4.

5.7.3.2 Lamb-Oseen

If the Lamb-Oseen method is used (***RegFunction=[2]***), the viscous core correction is given by Eq. 5.18.

$$F_\nu = \left[1 - \exp(-\frac{\rho^2}{r_c^2})\right] \tag{5.18}$$

5.7.3.3 Vatistas

If the Vatistas method is used (***RegFunction=[3]***), the viscous core correction is given by Eq. 5.19.

$$F_\nu = \frac{\rho^2}{(\rho^{2n} + r_c^{2n})^{1/n}} = \frac{(\rho/r_c)^2}{(1 + (\rho/r_c)^{2n})^{1/n}} \tag{5.19}$$

Here, ρ is the distance from a vortex segment to an arbitrary point (Abedi). Research from rotorcraft applications suggests a value of $n = 2$, which is used in this work (Bagai and Leishman).

5.7.3.4 Denominator Offset/Cut-Off

If the denominator offfset method is used (***RegFunction=[4]***), the viscous core correction is given by Eq. 5.20

$$\vec{v}(\vec{x}) = \frac{\Gamma}{4\pi} \frac{(r_1 + r_2)}{r_1 r_2 (r_1 r_2 + \vec{r}_1 \cdot \vec{r}_2) + r_c^2 r_0^2} \vec{r}_1 \times \vec{r}_2 \tag{5.20}$$

Here, the singularity is removed by introducing an additive factor in the denominator of Eq. 5.14, proportional to the filament length r_0. In this case, $F_\nu = 1$. This method is found in the work of van Garrel (Garrel).

5.7.4 Time Evolution of the Regularization Parameter—Core Spreading Method

There are four available methods by which the regularization parameter may evolve with time: 1) constant value, 2) stretching, 3) wake age, and 4) stretching and wake age. The three latter methods blend the notions of viscous diffusion and regularization. The notation r_{c0} used in this section corresponds to input file parameter value ***WakeRegParam***.

5.7.4.1 Constant

If a constant value is selected (***WakeRegMethod=[1]***), the value of r_c remains unchanged for all Lagrangian markers throughout the simulation and is taken as the value given with the parameter ***WakeRegParam*** in meters.

$$r_c(\zeta) = r_{c0} \tag{5.21}$$

Here, ζ is the vortex wake age, measured from its emission time.

5.7.4.2 Stretching

If the stretching method is selected (***WakeRegMethod=[2]***), the viscous core radius is modeled by Eq. 5.22.

$$r_c(\zeta,\varepsilon) = r_{c0}(1+\varepsilon)^{-1} \tag{5.22}$$

$$\varepsilon = \frac{\Delta l}{l} \tag{5.23}$$

Here, ε is the vortex-filament strain, l is the filament length, and Δl is the change of length between two time steps. The integral in Eq. 5.22 represents strain effects.

5.7.4.3 Wake Age / Core-Spreading

If the wake age method is selected (***WakeRegMethod=[3]***), the viscous core radius is modeled by Eq. 5.24.

$$r_c(\zeta) = \sqrt{r_{c0}^2 + 4\alpha\delta\nu\zeta} \tag{5.24}$$

where $\alpha = 1.25643$, ν is kinematic viscosity, and δ is a viscous diffusion parameter (typically between 1 and 1,000). The parameter δ is provided in the input file as ***CoreSpreadEddyVisc***. Here, the term $4\alpha\delta\nu\zeta$ accounts for viscous effects as the wake propagates downstream. The higher the background turbulence, the more diffusion of the vorticity with time, and the higher the value of δ should be. This method partially accounts for viscous diffusion of the vorticity while neglecting the interaction between the wake vorticity itself or between the wake vorticity and the background flow. It is often referred to as the core-spreading method. Setting ***DiffusionMethod=[1]*** is the same as using the wake age method (***WakeRegMethod=[3]***).

5.7.4.4 Stretching and Wake Age

If the stretching and wake-age method is selected (***WakeRegMethod=[4]***), the viscous core radius is modeled by Eq. 5.25.

$$r_c(\zeta,\varepsilon) = \sqrt{r_{c0}^2 + 4\alpha\delta\nu\zeta\left(1+\varepsilon\right)^{-1}} \tag{5.25}$$

5.8 Diffusion

The viscous-splitting assumption is used to solve for the convection and diffusion of the vorticity separately. The diffusion term $\nu\Delta\vec{\omega}$ represents molecular diffusion. This term allows for viscous connection of vorticity lines. Also, turbulent flows will diffuse the vorticity in a similar manner based on a turbulent eddy viscosity.

The parameter ***DiffusionMethod*** is used to switch between viscous diffusion methods. Currently, only the core-spreading method is implemented. The method is described in Section 5.7.4 since it is equivalent to the increase of the regularization parameter with the wake age.

6 State-Space Representation and Integration with OpenFAST

6.1 State, Constraint, Input, and Output Variables

The OLAF module has been integrated into the latest version of OpenFAST via *AeroDyn15*, following the OpenFAST modularization framework (Jonkman; Sprague, Jonkman, and Jonkman). To follow the OpenFAST framework, the vortex code is written as a module, and its formulation comprises state, constraint, and output equations. The data manipulated by the module include the following vectors: constant parameters, $\vec{p}$; inputs, $\vec{u}$; constraint states, $\vec{z}$; states, $\vec{x}$; and outputs, $\vec{y}$. The vectors are defined as follows:

- Parameters, $\vec{p}$ — a set of internal system values that are independent of the states and inputs. The parameters can be fully defined at initialization and characterize the system state and output equations.
- Inputs, $\vec{u}$ — a set of values supplied to the module that, along with the states, are needed to calculate future states and the system output.
- Constraint states, $\vec{z}$ — algebraic variables that are calculated using a nonlinear solver, based on values from the current time step.
- States, $\vec{x}$ — a set of internal values of the module. They are influenced by the inputs and used to calculate future state values and output. Continuous states are employed, meaning that the states are differentiable in time and characterized by continuous time-differential equations.
- Outputs, $\vec{y}$ — a set of values calculated and returned by the module that depend on the states, inputs, and/or parameters through output equations.

The parameters of the vortex code include:

- Fluid characteristics: kinematic viscosity, ν.
- Airfoil characteristics: chord c and polar data—$C_l(\alpha)$, $C_d(\alpha)$, $C_m(\alpha)$.
- Algorithmic methods and parameters for, e.g., regularization, viscous diffusion, discretization, wake geometry, and acceleration.

The inputs of the vortex code are:

- Position, orientation, translational velocity, and rotational velocity of the different nodes of the lifting lines ($\vec{r}_{ll}$, Λ_{ll}, $\dot{\vec{r}}_{ll}$, and $\vec{\omega}_{ll}$, respectively), gathered into the vector, $\vec{x}_{\text{elast},ll}$, for conciseness. These quantities are handled using the mesh-mapping functionality and data structure of OpenFAST.
- Disturbed velocity field at requested locations, written $\vec{V}_0 = [\vec{V}_{0,ll}, \vec{V}_{0,m}]$. Locations are requested for lifting-line points, $\vec{r}_{ll}$, and Lagrangian markers, $\vec{r}_m$. Based on the parameters, this disturbed velocity field may contain the following influences: freestream, shear, veer, turbulence, tower, and nacelle disturbance. The locations where the velocity field is requested are typically the location of the Lagrangian markers.

The constraint state is:

- The circulation intensity along the lifting lines, Γ_{ll}.

The continuous states are:

- The position of the Lagrangian markers, $\vec{r}_m$
- The vorticity associated with each vortex element, $\vec{\omega}_e$. For a projection of the vorticity onto vortex segments, this corresponds to the circulation, $\vec{\Gamma}_e$. For each segment, $\vec{\Gamma}_e = \Gamma_e \vec{dl}_e = \vec{\omega}_e dV_e$, were $\vec{dl}_e$ and dV_e are the vortex segment length and its equivalent vortex volume, respectively.

The outputs are [1]:

- The induced velocity at the lifting-line nodes, $\vec{v}_{i,ll}$
- The locations where the undisturbed wind is computed, $\vec{r}_r$ (typically $\vec{r}_r = [\vec{r}_{ll}, \vec{r}_m]$).

6.2 State, Constraint, and Output Equations

An overview of the states, constraints, and output equations is given here. More details are provided in Section 5. The constraint equation is used to determine the circulation distribution along the span of each lifting line. For the van Garrel method, this circulation is a function of the angle of attack along the blade and the airfoil coefficients. The angle of attack at a given lifting-line node is a function of the undisturbed velocity, $\vec{v}_{0,ll}$, and the vorticity-induced velocity, $\vec{v}_{i,ll}$, at that point. Part of the induced velocity is caused by the vorticity being shed and trailed at the current time step, which in turn is a function of the circulation distribution along the lifting line. This constraint equation may be written as:

$$\vec{Z} = \vec{0} = \vec{\Gamma}_{ll} - \vec{\Gamma}_p\left(\vec{\alpha}(\vec{x},\vec{u}),\vec{p}\right) \tag{6.1}$$

where $\vec{\Gamma}_p$ is the function that returns the circulation along the blade span, according to one of the methods presented in Section 5.3.

The state equation specifies the time evolution of the vorticity and the convection of the Lagrangian markers.

$$\frac{d\vec{\omega}_e}{dt} = \left[(\vec{\omega}\cdot\nabla)\vec{v} + \nu\nabla^2\vec{\omega}\right]_e \tag{6.2}$$

$$\frac{d\vec{r}_m}{dt} = \vec{V}(\vec{r}_m) = \vec{V}_0(\vec{r}_m) + \vec{v}_\omega(\vec{r}_m) = \vec{V}_0(\vec{r}_m) + \vec{V}_\omega(\vec{r}_m,\vec{r}_m,\vec{\omega}) \tag{6.3}$$

Here,

- $\vec{v}_\omega$ is the velocity induced by the vorticity in the domain;
- $\vec{V}_\omega(\vec{r},\vec{r}_m,\vec{\omega})$ is the function that computes this induced velocity at a given point, $\vec{r}$, based on the location of the Lagrangian markers and the intensity of the vortex elements;
- the subscript e indicates that a quantity is applied to an element; and
- the vorticity, $\vec{\omega}$, is recovered from the vorticity of the vortex elements by means of discrete convolutions.

For vortex-segment simulations, the viscous-splitting algorithm is used, and the convection step (Eq. 6.3) is the primary state equation being solved for. Vorticity stretching is automatically accounted for and the diffusion is performed *a posteriori*. The velocity function, $\vec{V}_\omega$, uses the Biot-Savart law. The output equation is:

$$\vec{y}_1 = \vec{v}_{i,ll} = \vec{V}_\omega(\vec{r}_{ll},\vec{r}_m,\vec{\omega}) \tag{6.4}$$

$$\vec{y}_2 = \vec{r}_r \tag{6.5}$$

6.3 Integration with AeroDyn15

The vortex code has been integrated as a submodule of the aerodynamic module of OpenFAST, *AeroDyn15*. The data workflow between the different modules and submodules of OpenFAST is illustrated in Figure 4. AeroDyn inputs such as BEM options (e.g., tip-loss factor), skew model, and dynamic inflow are discarded when the vortex code is used. The environmental conditions, tower shadow, and dynamic stall model options are used. This integration required a restructuring of the *AeroDyn15* module to isolate the parts of the code related to tower shadow modeling, induction computation, lifting-line-forces computations, and dynamic stall. The dynamic stall model is adapted when used in conjunction with the vortex code to ensure the effect of shed vorticity is not accounted for twice. The interface between *AeroDyn15* and the inflow module, *InflowWind*, was accommodated to include the additionally requested points by the vortex code.

[1] The loads on the lifting line are not an output of the vortex code; their calculation is handled by a separate submodule of *AeroDyn*.

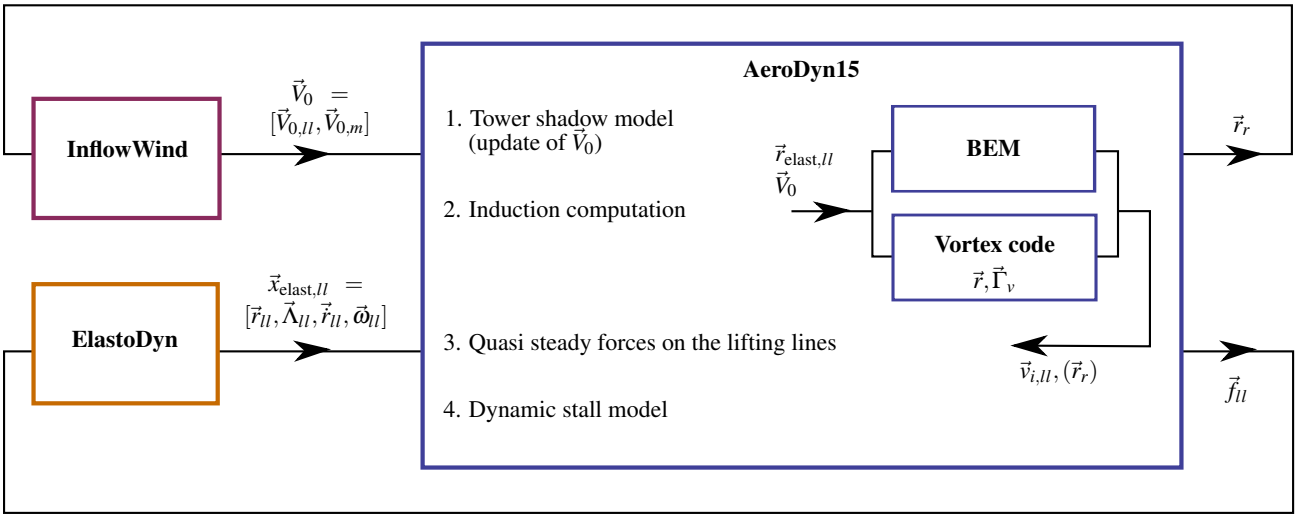


Figure 4. OpenFAST-OLAF code integration workflow

7 Future Work

This first implementation phase focused on single-turbine capabilities, fulfilling the basic requirements for the design of large and novel rotor concepts. Future development will focus on the implementation of features enabling multiple-turbine simulations on medium-to-large-scale computational clusters, as well as the reduction of computational time. This may be achieved using tree techniques such as the fast multipole method. Further algorithmic options, such as vortex amalgamation in the far wake, will be considered to speed up the simulation. The framework presented in this manual is compatible with grid-free or grid-based vortex particle formulations. Such particle-based implementations will also be envisaged in the future. Further validation of the code against measurements and higher-order tools will be pursued. Applications to cases known to be challenging for the BEM algorithm will also be investigated, such as highly flexible rotors, offshore floating turbines, small-scale wind farms, multiple-rotor turbines, or kites.

The following list contains future work on OLAF software:

- Lagrangian particles
- Multiple turbines, integration into FAST.Farm
- Code speed-up
- Dedicated dynamic stall model

Bibliography

Abedi, H. *Development of Vortex Filament Method for Wind Power Aerodynamics*. PhD thesis, Chalmers University of Technology, 2016.

Ananthan, S., J. G. Leishman, and M. Ramasamy. "The Role of Filament Stretching in the Free-Vortex Modeling of Rotor Wakes". In *58th Annual Forum and Technology Display of the American Helicopter Society International*. Montreal, Canada, 2002.

Bagai, A., and J. G. Leishman. "Flow Visualization of Compressible Vortex Structures Using Density Gradient Techniques". *Experiments in Fluids* 15, no. 6 (1993): 431–442.

– . "Rotor Free-Wake Modeling using a Pseudo-Implicit Technique Including Comparisons with Experimental Data". In *50th Annual Forum of the American Helicopter Society*. Washington, D.C., 1994.

Branlard, E. *Wind Turbine Aerodynamics and Vorticity-Based Methods: Fundamentals and Recent Applications*. Springer International Publishing, 2017. ISBN: 978-3-319-55163-0. https://doi.org/10.1007/978-3-319-55164-7.

Branlard, E., et al. "Aeroelastic large eddy simulations using vortex methods: unfrozen turbulent and sheared inflow". *Journal of Physics: Conference Series (Online)* 625 (2015). ISSN: 1742-6596. https://doi.org/10.1088/1742-6596/625/1/012019.

Garrel, A. van. *Development of a Wind Turbine Aerodynamics Simulation Module*. Tech. rep. ECN-C–03-079. ECN, 2003.

Gupta, S. *Development of a Time-Accurate Viscous Lagrangian Vortex Wake Model for Wind Turbine Applications*. PhD thesis, Univeristy of Maryland, 2006.

Gupta, S., and J. G. Leishman. "Free-Vortex Filament Methods for the Analysis of Helicopter Rotor Wakes". *Journal of Aircraft* 39, no. 5 (2002): 759–775.

Hansen, M. O. L. *Aerodynamics of Wind Turbines*. London; Sterling, VA: Earthscan, 2008.

Jonkman, J. *The New Modularization Framework for the FAST Wind Turbine CAE Tool*. Technical report NREL/CP-5000-57228. National Renewable Energy Laboratory, 2013.

Kerwin, J. *Lecture Notes Hydrofoil and propellers*. Tech. rep. M.I.T., 2000.

Leishman, J. *Principles of Helicopter Aerodynamics*. Cambridge, MA: Cambridge Univ. Press, 2006.

Leishman, J. G., M. J. Bhagwat, and A. Bagai. "Free-Vortex Filament Methods for the Analysis of Helicopter Rotor Wakes". *Journal of Aircraft* 39, no. 5 (2002): 759–775.

Papadakis, G. *Development of a hybrid compressible vortex particle method and application to external problems including helicopter flows*. PhD thesis, National Technical University of Athens, 2014.

Rankine, W. J. M. *Manual of Applied Mechanics*. London: Griffen Co., 1858.

Ribera, M. *Helicopter Flight Dynamics Simulation with a Time-Accurate Free-Vortex Wake Model*. PhD thesis, University of Maryland, 2007.

Rosenhead, L. "The Formation of Vortices from a Surface of Discontinuity". *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 134, no. 823 (1931): 170–192. ISSN: 09501207. http://www.jstor.org/stable/95835.

Scully, M. P. *Computation of Helicopter Rotor Wake Geometry and Its Influence on Rotor Harmonic Airloads*. PhD thesis, Massachusetts Institute of Technology, 1975.

Sessarego, M., et al. "Development of an aeroelastic code based on three-dimensional viscous-inviscid method for wind turbine computations". *Wind Energy* 20, no. 7 (2017): 1145–1170. https://doi.org/10.1002/we.2085.

Sprague, Michael A., Jason M. Jonkman, and Bonnie J. Jonkman. *FAST Modular Framework for Wind Turbine Simulation: New Algorithms and Numerical Examples*. Tech. rep. NREL/CP-2C00-63203. National Renewable Energy Laboratory, 2015.

Vatistas, G. H., V. Koezel, and W. C. Mih. "A Simpler Model for Concentrated Vortices". *Experiments in Fluids* 11, no. 1 (1991): 73–76.

Voutsinas, S. G. "Vortex methods in aeronautics: how to make things work". *International Journal of Computational Fluid Dynamics* (2006).

Weissinger, J. *The Lift Distribution of Swept-Back Wings*. Technical report TM 1120. NACA, 1947.
Winckelmans, G. S., and A. Leonard. “Contributions to vortex particle methods for the computation of 3-dimensional incompressible unsteady flows”. *Journal Of Computational Physics* 109, no. 2 (1993): 247–273. ISSN: 00219991, 10902716.

A OLAF Primary Input File

The symbol ↪ in the following text indicates a continuation of the previous line, it should not be implemented as a new line in the actual input file. When a default value is available, `default` may be used instead of the value.

```
---------------------------- FREE WAKE INPUT FILE -------------------------------------------
Free wake input file for the BAR turbine
---------------------------- GENERAL OPTIONS ------------------------------------------------
5      IntMethod           Integration method {5: Forward Euler 1st order, default: 5} (switch)
0.2    DTfvw               Time interval for wake propagation. {default: dtaero} (s)
5      FreeWakeStart       Time when wake is free. (-) value = always free. {default: 0.0} (s)
2.0    FullCircStart       Time at which full circulation is reached. {default: 0.0} (s)
---------------------------- CIRCULATION SPECIFICATIONS -------------------------------------
1      CircSolvingMethod   Circulation solving method {1: Cl-Based, 2: No-Flow Through, 3:
                            ↪ Prescribed, default: 1 }(switch)
0.01   CircSolvConvCrit    Convergence criteria {default: 0.001} [only if CircSolvingMethod=1] (-)
0.1    CircSolvRelaxation  Relaxation factor {default: 0.1} [only if CircSolvingMethod=1] (-)
30     CircSolvMaxIter     Maximum number of iterations for circulation solving {default: 30} (-)
 "NA"  PrescribedCircFile  File containing prescribed circulation [only if CircSolvingMethod=3]
                            ↪ (quoted string)
=========================================================================================
---------------------------- WAKE OPTIONS ---------------------------------------------------
------------------- WAKE EXTENT AND DISCRETIZATION ----------------------------------------
50     nNWPanel            Number of near-wake panels (-)
500    WakeLength          Total wake distance [integer] (number of time steps)
400    FreeWakeLength      Wake length that is free [integer] (number of time steps) {default:
                            ↪ WakeLength}
False  FWShedVorticity     Include shed vorticity in the far wake   {default: False}
------------------- WAKE REGULARIZATIONS AND DIFFUSION ------------------------------------
0      DiffusionMethod     Diffusion method to account for viscous effects {0: None, 1: Core
                            ↪ Spreading, "default": 0}
0      RegDeterMethod      Method to determine the regularization parameters {0:  Manual, 1:
                            ↪ Optimized, default: 0 }
2      RegFunction         Viscous diffusion function {0: None, 1: Rankine, 2: LambOseen, 3:
                            ↪ Vatistas, 4: Denominator, "default": 3} (switch)
0      WakeRegMethod       Wake regularization method {1: Constant, 2: Stretching, 3: Age,
                            ↪ default: 1} (switch)
2.0    WakeRegFactor       Wake regularization factor (m)
2.0    WingRegFactor       Wing regularization factor (m)
100    CoreSpreadEddyVisc  Eddy viscosity in core spreading methods, typical values 1-1000
------------------- WAKE TREATMENT OPTIONS ------------------------------------------------
False  TwrShadowOnWake     Include tower flow disturbance effects on wake convection
                            ↪ {default:false} [only if TwrPotent or TwrShadow]
0      ShearModel          Shear Model {0: No treatment, 1: Mirrored vorticity, default: 0}
------------------- SPEEDUP OPTIONS -------------------------------------------------------
2      VelocityMethod      Method to determine the velocity {1:Biot-Savart Segment, 2:Particle
                            ↪ tree, default: 1}
1.5    TreeBranchFactor    Branch radius fraction above which a multipole calculation is used
                            ↪ {default: 2.0} [only if VelocityMethod=2]
1      PartPerSegment      Number of particles per segment [only if VelocityMethod=2]
=========================================================================================
---------------------------- OUTPUT OPTIONS  ------------------------------------------------
1      WrVTk               Outputs Visualization Toolkit (VTK) (independent of .fst option) {0:
                            ↪ NoVTK, 1: Write VTK}
1      nVTKBlades          Number of blades for which VTK files are exported {0: No VTK per blade,
                            ↪ n: VTK for blade 1 to n} (-)
2      VTKCoord            Coordinate system used for VTK export. {1: Global, 2: Hub, "default": 1}
default VTK_fps            Frame rate for VTK output (frames per second) {"all" for all glue code
                            ↪ timesteps, "default" for all FVW timesteps} [used only if WrVTK=1]
-----------------------------------------------------------------------------------------
```

B Prescribed Circulation Input File

```
r/R [-],      Gamma [m²/s]
0.048488,     0.000000
0.087326,     0.442312
0.126163,     6.909277
0.165000,     23.678557
0.203837,     55.650700
0.242674,     74.091529
0.281512,     84.205843
0.320349,     88.740429
0.359186,     89.730814
0.398023,     88.568114
0.436860,     87.114743
0.475698,     86.110557
0.514535,     85.705529
0.553372,     85.215829
0.592209,     84.547371
0.631047,     83.774329
0.669884,     82.889157
0.708721,     81.635600
0.747558,     79.788700
0.786395,     77.195200
0.825233,     73.765100
0.864070,     69.275900
0.902907,     62.965400
0.941744,     53.603300
0.980581,     39.854000
```

C OLAF List of Output Channels

This is a list of all possible output parameters from the OLAF module. The names are grouped by meaning, but can be ordered in the OUTPUTS section of the *AeroDyn15* primary input file, as the user sees fit. $N\beta$ refers to output node, β, where β is a number in the range [1,9], corresponding to entry, β, in the ***OutNd*** list. $B\alpha$ is prefixed to each output name, where α is a number in the range [1,3], corresponding to the blade number.

Table 1. Available OLAF Output Channels

Channel Name(s)	Units	Description
$B\alpha N\beta Gam$	m^2/s	Circulation along the blade